Formularia

Francisco Javier Almazán Navajo
Sergio Castilla Rufo
Mario Espino Sanz

Facultad de Informática
Universidad Complutense de Madrid

Proyecto supervisado por:

Juan Carlos Fabero

Sistemas Informáticos 2011/2012

Se autoriza a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria como el código, la documentación y/o el prototipo desarrollado.

Francisco Javier Almazan Navajo Sergio Castilla Rufo Mario Espino Sanz

Resumen

Formularia es una aplicación para dispositivos móviles Android que tengan instalada una versión 2.2 o superior. Formularia permite la cumplimentación de formularios, previamente programados, con una serie de restricciones para cada campo, facilitando el correcto rellenado de los mismos.

Palabras clave: Android, Rellenado de formulario, XML, XSLT, Servidor externo.

Abstract

Formularia is an application for Android mobile device with a 2.2 firmware version or later. Formularia allows application forms filling, previously programmed, with a series of restrictions for each field, helping to complete correctly each one of them

Keywords: Android, Application form filling, XML, XSLT, External server.

Índice General

1 Prólogo **1**

1.1 Introducción 1

1.2 Objetivos del Proyecto 1

1.3 Estructura del documento 3

2 Revisión de conceptos y tecnologías **5**

2.1 Introducción 5

2.2 Motivaciones 5

2.3 Otras tecnologías disponibles 6

2.3.1 Adobe Acrobat Mobile PDF 6

2.3.2 Google Docs 6

2.4 Lenguaje de marcado 8

2.4.1 XML 8

2.4.2 XSLT 10

2.5 Bases de datos 11

2.5.1 MySql 11

2.5.2 Sqlite 12

2.6 Servidores 13

2.6.1 Apache 13

2.6.2 Tomcat 14

3 Formularia **15**

3.1 Introducción 15

3.2 La aplicación 16

3.2.1 Vistas 16

3.2.1.1 Login . . . 16
3.2.1.2 Pantalla Principal . . . 19
3.2.1.3 Nuevo Formulario . . . 20
3.2.1.4 Abrir Formulario . . . 21
3.2.1.5 Vista Previa . . . 22
3.2.1.6 Rellenar Formulario . . . 23
3.2.1.7 Firmar . . . 28
3.2.1.8 Preferencias . . . 28
3.2.2 Edición de formularios . . . 29
3.2.3 Seguridad . . . 31
3.2.4 Conectividad . . . 32
3.2.4.1 Formulario-modelo . . . 32
3.2.4.2 Formulario-usuario . . . 34
3.3 La base de datos . . . 36
3.3.1 Base de datos interna . . . 36
3.3.2 Base de datos externa . . . 36
3.4 Programación de formularios . . . 37

4 Desarrollo e Implementación 39
4.1 Introducción . . . 39
4.2 Método de Desarrollo . . . 39
4.3 Estructura de la base de datos . . . 42
4.3.1 Base de datos interna . . . 42
4.3.2 Base de datos externa . . . 45
4.3.3 Posibles mejoras . . . 49
4.4 Estructura de la aplicación . . . 49
4.4.1 Views . . . 49
4.4.2 Seguridad . . . 58
4.4.3 Modelo . . . 58
4.4.3.1 Items . . . 59
4.4.3.2 Restricciones . . . 60
4.4.3.3 Posibles mejoras . . . 61
4.4.4 XMLHandler . . . 61

4.5 Formularios 61
4.5.1 XML 61
4.5.2 XSLT 64

5 Conclusiones y Trabajo Futuro **67**
5.1 Conclusiones 67
5.2 Trabajo futuro 68

6 Referencias **69**

1

Prólogo

1.1 Introducción

El tema abordado en este proyecto de Sistemas Informáticos es el desarrollo de una aplicación para la cumplimentación de formularios, ya sea para una empresa o para uso público, en dispositivos móviles como PDAs, tablets o teléfonos multimedia con sistema operativo Android.

Figure 1.1: Formularia - Icono de la aplicación.

1.2 Objetivos del Proyecto

El principal objetivo de este proyecto de Sistemas Informáticos es la creación de una aplicación para la cumplimentación de formularios en dispositivos móviles. La aplicación debe permitir:

- La creación de formularios por parte del administrador del sistema de forma que

se puedan diferenciar por un lado los datos y estructura y por otro el formato de visualización de los mismos.

- Estos formularios tienen que tener una estructura definida específicamente para cada campo del mismo. Por ejemplo: si un campo es para un email sólo puede rellenarse con una dirección email. Si el campo es un mes sólo se podría rellenar con algún mes del año.

- Estos formularios deben poder ser almacenados en una base de datos externa, a la que se pueda acceder desde el dispositivo móvil y así poder descargar los formularios a una base de datos interna del propio móvil.

- Los formularios que estén almacenados en el dispositivo móvil pueden ser subidos y/o actualizados a la base de datos externa. Así mismo los formularios tienen que poder ser editables ya estén guardados en la base de datos interna o en la externa.

- La aplicación debe poder contar con diferentes usuarios. Estos usuarios pueden acceder a unos formularios base determinados (llamados formularios-modelo). Cada usuario puede acceder a todos los formularios rellenados por él mismo (formularios-usuario).

- Un usuario puede pertenecer a ninguno o varios grupos de trabajo. Estos grupos tienen diferentes permisos de acceso a los formularios-modelo.

- La aplicación debe tener un modo sin conexión con el que el usuario pueda trabajar con los formularios-modelo que previamente haya dispuesto para su uso sin conexión. Estos formularios se guardarán sólo en la base de datos interna del móvil. Los formularios-usuario (formularios-modelo rellenados) que cree este modo sin conexión tienen que poder ser reclamados por aquellos usuarios que tengan acceso a los formulario-modelos de los que derivan.

- Tiene que contemplar una opción de seguridad que sólo permita trabajar de forma remota con el servidor externo y, así, no guardar nada en el dispositivo.

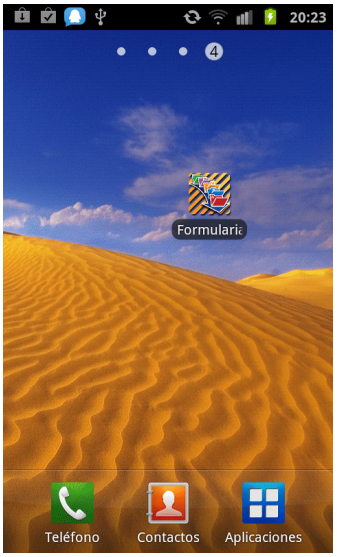

Figure 1.2: Formularia - La aplicación Formularia en el escritorio de un dispositivo Android.

1.3 Estructura del documento

La estructura de la memoria es la siguiente:

- En la segunda sección se realiza una introducción a las motivaciones y tecnologías más relevantes en el proyecto, así como una breve introducción a las herramientas existentes y las utilizadas en el proyecto.
- En la tercera sección se describe la aplicación Formularia, presentando sus distintas pantallas y las distintas funcionalidades dentro de la aplicación, así como las bases de datos necesarias. También se presenta la creación de formularios por parte del administrador.
- En la cuarta sección se detalla la arquitectura e implementación de la aplicación Formularia, la documentación necesaria para la implementación de los formularios y la información para la creación de la base de datos externa. Y se explicará cómo crear formularios-modelo personalizados.
- Por último, en la quinta sección se presentan las principales conclusiones obtenidas tras la finalización del proyecto, así como posibles extensiones y trabajos futuros.

2

Revisión de conceptos y tecnologías

2.1 Introducción

Hoy en día las nuevas tecnologías están avanzando rápidamente y se están integrando cada vez más en la vida cotidiana de las personas. Tanto en el trabajo como en el tiempo de ocio hay tecnologías como Internet, la telefonía móvil y otras que se han hecho indispensables. Cada vez más cosas son complementadas, si no realizadas totalmente, con las nuevas tecnologías haciendo que la realización de estas se simplifique en gran medida y se abran a la vez nuevas oportunidades que antes, ya fuera por el tiempo o la complejidad de hacerlo de otra forma, no eran posibles. En este marco surge la idea de Formularia como una herramienta para facilitar la labor de cumplimentación de los formularios.

2.2 Motivaciones

La cumplimentación de formularios es a día de hoy un hecho cotidiano. Ya sea para facilitar información o para realizar un trabajo, independientemente del formato, la introducción de información es necesaria y en cada caso tiene a seguir unas reglas. Con la motivación de aunar todas esas necesidades y darles solución nace Formularia.

En un ejemplo práctico se propone un inspector de obra. Este inspector al comenzar una inspección deberá rellenar un formulario estándar cumplimentando los campos de una manera determinada de forma que cuando termine la inspección el formulario será el registro de ésta. Aún así, el inspector puede no tener tiempo para terminar la inspección, por lo que el formulario quedará incompleto pero aun así deberá ser guardado para terminarlo más adelante. Una vez terminado este formulario deberá ser remitido a un superior o archivado para su posterior análisis.

Esta aplicación no sólo cubre este caso. El caso de un ciudadano haciendo una petición de una prestación o el formulario de petición de días libres o vacaciones, además de otros muchos más casos. Esta aplicación trata de cubrir todos estos casos e implementar una solución eficaz para facilitar todo tipo de trabajos abarcando un área que actualmente no se encuentra cubierta.

2.3 Otras tecnologías disponibles

En el mercado hay otras alternativas que cubren uno o varios aspectos de los propuestos por la aplicación Formularia.

2.3.1 Adobe Acrobat Mobile PDF

Adobe tiene a disposición su herramienta Acrobat. Con esta herramienta se permite la creación de formularios en pdf con campos interactivos además de incluir imágenes y vídeos. Además en sus últimas versiones también se pueden incluir animaciones según gps, bases de datos y otros. A día de hoy es una herramienta muy completa pero no tiene una versión para dispositivos móviles en el apartado de la edición y realización de formularios.

2.3.2 Google Docs

En los últimos años y tras continuas actualizaciones el conjunto de soluciones de ofimática de Google se ha destacado por aceptar muchos tipos de documentos que

Figure 2.1: Adobe Acrobat Mobile - Rellenando formularios en Acrobat Reader Mobile.

pueden ser editados por varios usuarios a la vez en la nube. Una de estas actualizaciones fue la de los formularios de Google, basados en HTML, con 60 temas diferentes y 7 tipos de preguntas. Al igual que con la opción anterior es una herramienta muy completa pero la aplicación para dispositivos móviles no permite hoy en día ni la creación ni la edición de formularios.

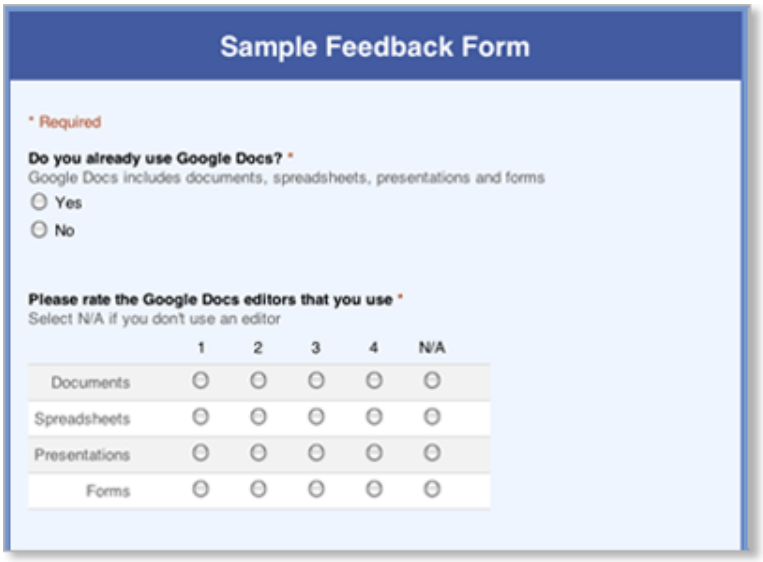

Figure 2.2: Google Docs - Ejemplo de diseño de un formulario con Google Docs.

2.4 Lenguaje de marcado

Los lenguajes de marcado consisten en una manera de codificar un documento de tal forma que este contenga etiquetas o marcas que proporcionan información acerca de la estructura del texto o de su presentación. Algunos ejemplos de este tipo de lenguajes son: SGML, HTML o XML.

A continuación se pasa a describir de forma concreta XML, por ser el utilizado en el proyecto como soporte de almacenaje de la parte de los datos de los formularios.

2.4.1 XML

XML (eXtensible Markup Language)[David Hunter 2007] es un metalenguaje extensible de marcado. Ha sido desarrollado por el World Wide Web Consortium (W3C).

La principal ventaja de XML es su gran versatilidad. XML es una herramienta utilizada en todo tipo de aplicaciones y para multitud de finalidades. Esto incluye desde la comunicación de dos ordenadores a través de una red o Internet, hasta la persistencia de datos, pasando por la definición de la estructura de un tipo de documento.

XML surgió como una simplificación y adaptación del SGML (Standard Generalized Markup Language)[Charles F.Goldfarb 1998]. El lenguaje HTML está definido en términos del SGML, ya que la normalización de XML fue posterior al diseño del lenguaje de marcas HTML.

Además, XML cuenta con el concepto de "documento bien formado" y "validez" como elementos separados:

- Documento bien formado: es aquel documento que cumple todas las definiciones básicas del formato y puede, por tanto, ser analizado correctamente por cualquier analizador sintáctico que cumpla con los estándares.

- Documento válido: es aquel documento que además de ser un documento bien formado, es decir, cumple con todas las definiciones básicas del formato y por ello puede ser analizado correctamente, semánticamente es correcto, o aquello que describe tiene sentido. Para que la comprobación de validez se pueda realizar, se

necesitará un modelo gramatical del documento, que exprese las relaciones entre los diferentes elementos del documento XML, y que establezca unos límites para los valores de los atributos. Como un ejemplo, un documento podrá considerarse inválido si guarda en un campo fecha un valor de "31 de febrero de 2010". La gramática documental sería la encargada de fijar los límites de este campo "fecha". XML incluye un formalismo de gramática documental denominado DTD (Document Type Definition). No obstante, se han propuesto otros formalismos con mayor poder expresivo, como XML Schema [Eric van der Vlist 2002].

```
<?xml version="1.0" encoding="UTF-8" ?>
- <_Catalog entry_count="4">
  - <Entry display_name="From" type="string" nlp_usage="Sender"
    is_viewed="true" is_categories="false" is_link="false">
      <![CDATA[ From ]]>
    </Entry>
  - <Entry display_name="Message" type="string" nlp_usage="Body"
    is_viewed="true" is_categories="false" is_link="false">
      <![CDATA[ Message ]]>
    </Entry>
  - <Entry display_name="Subject" type="string" nlp_usage="Subject"
    is_viewed="true" is_categories="false" is_link="false">
      <![CDATA[ Subject ]]>
    </Entry>
  - <Entry display_name="_Categories" type="classification"
    nlp_usage="" is_viewed="true" is_categories="true" is_link="false">
      <![CDATA[ _Categories ]]>
    </Entry>
  </_Catalog>
```

Figure 2.3: XML - Ejemplo de como codificar un XML

Las partes de un documento XML están bien definidas y son:

- Prólogo: contiene información acerca del documento XML. Describe la versión de XML, el tipo de documento, la codificación utilizada, y otro tipo de metainformación. Es opcional para que un documento esté bien formado.

- Cuerpo: el cuerpo debe contener un único elemento raíz para que el documento esté bien formado. Es obligatorio para que un documento esté bien formado.
- Elementos: los elementos pueden contener: elementos, caracteres o ambos.
- Atributos: los elementos pueden contener atributos, que son una manera de añadir características o propiedades a los elementos de un documento. Deben ir entre comillas para que un documento esté bien formado.
- Entidades predefinidas: entidades para representar caracteres especiales para que el analizador sintáctico de XML no los interprete como marcado. Por ejemplo, &amp es el símbolo &.
- Secciones CDATA: es una construcción XML para especificar datos utilizando cualquier carácter sin que se interprete como marcado XML. Así se consigue que caracteres especiales no rompan la estructura XML.
- Comentarios: comentarios a título informativo por parte del diseñador del XML, que el procesador de XML ignorará.

Además de todo lo expuesto, XML cuenta con una gran facilidad de transformación. XSLT, por ejemplo, es otro estándar desarrollado por el World Wide Web Consortium (W3C) que permite muy fácilmente transformar documentos XML y aplicarles estilos [Doug Tidwell 2008] pudiendo conseguir un documento HTML a partir de un XML. De esta manera, se separa el contenido de la presentación de una manera muy natural.

En general, XML cuenta con un ecosistema de herramientas y lenguajes anexos muy potentes que hacen que trabajar con este lenguaje de marcado sea lo idóneo en un gran número de ocasiones.

2.4.2 XSLT

XSLT (eXtensible Stylesheet Language Transformation) [James Clark y Michael Kay 2007] es un estándar de la organización World Wide Web Consortium (W3C). Este lenguaje fue ideado para poder transformar documentos XML en otros sin modificar el

documento original. XSLT realizan la transformación del documento utilizando una o varias reglas de plantilla creando un nuevo documento que tiene el contenido del XML original. Este documento de salida puede ser serializado en una sintaxis XML estándar o en otro formato, como HTML o texto plano. XSLT también puede ser usado para transformar un documento XML que obedezca a un DTD determinado en otro que sea diferente y conseguir que dos bases de datos en DTD puedan ser compatibles.

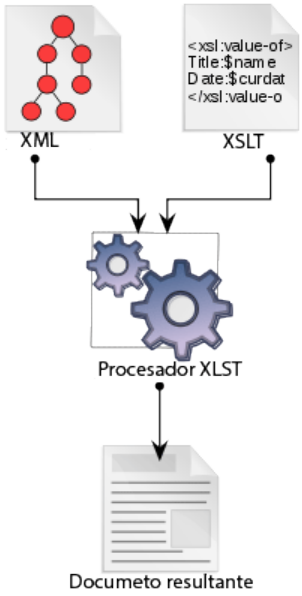

Figure 2.4: XSLT - Esquema de la utilización de XML junto con XSLT para formar un nuevo documento

2.5 Bases de datos

Las distintas bases de datos de la aplicación están implementadas en dos tecnologías diferentes: MySql y Sqlite.

2.5.1 MySql

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario desarrollado como software libre mayormente en el lenguaje ANSI C.

MySQL es un sistema de administración de bases de datos. Una base de datos es una colección estructurada de tablas que contienen datos. Para agregar, acceder y procesar datos guardados en un computador, se necesita un sistema gestor y administrador de bases de datos como MySQL Server. Los administradores de bases de datos juegan un papel central en computación, como aplicaciones independientes o como parte de otras aplicaciones.

Entre sus principales características podemos encontrar:

- Uso de multihilos mediante hilos del kernel.
- Completo soporte para operadores y funciones en cláusulas de selección filtrada.
- Completo soporte para cláusulas de agrupamiento y ordenado, soporte de funciones de agrupación
- Seguridad: ofrece un sistema seguro de contraseñas y privilegios mediante verificación basada en el equipo informático anfitrión y el tráfico de contraseñas está cifrado al conectarse a un servidor.
- Soporta gran cantidad de datos. MySQL Server tiene bases de datos de hasta 50 millones de registros.
- Los clientes se conectan al servidor MySQL usando una conexión TCP/IP en cualquier plataforma.
- MySQL contiene su propio paquete de pruebas de rendimiento proporcionado con el código fuente de la distribución de MySQL.

2.5.2 Sqlite

Sqlite es un sistema de gestión de base de datos relacional compatible con ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad en castellano) contenida en una pequeña biblioteca escrita en C. Este sistema es de dominio público y es por ello por lo que se incluye dentro de Android.

A diferencia de los sistemas de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina anfitriona. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción.

La biblioteca implementa la mayor parte del estándar SQL-92, incluyendo transacciones de base de datos atómicas, consistencia de base de datos, aislamiento, y durabilidad (ACID), disparadores y la mayor parte de las consultas complejas.

La versión utilizada en Android es la 3, que permite bases de datos de hasta dos terabytes de tamaño.

2.6 Servidores

El servidor externo, que sostiene la base de datos externa, se trata de un servidor Apache-Tomcat. Apache-Tomcat es un tándem entre dos servidores: un servidor http llamado Apache, y un servidor de aplicaciones web llamado Tomcat.

2.6.1 Apache

El servidor HTTP Apache es un servidor web HTTP de código abierto, para plataformas Unix (BSD,GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual.

Apache se usa principalmente para enviar páginas web estáticas y dinámicas en la red. Muchas aplicaciones web están diseñadas asumiendo como ambiente de implantación a Apache, o que utilizarán características propias de este servidor web.

Apache se usa para muchas otras tareas donde el contenido necesita ser puesto a disposición en una forma segura y confiable. Un ejemplo es al momento de compar-

tir archivos desde una computadora personal hacia Internet. Un usuario que tiene Apache instalado en su escritorio puede colocar arbitrariamente archivos en la raíz de documentos de Apache, desde donde pueden ser compartidos.

Los programadores de aplicaciones web a veces utilizan una versión local de Apache con el fin de previsualizar y probar código mientras éste es desarrollado.

La licencia de software bajo la cual el software de la fundación Apache es distribuido es una parte distintiva de la historia de Apache HTTP Server y de la comunidad de código abierto. La Licencia Apache permite la distribución de derivados de código abierto y cerrado a partir de su código fuente original.

2.6.2 Tomcat

Tomcat es un servidor web con soporte de servlets y JSPs. Tomcat incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache.

Tomcat puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

3

Formularia

3.1 Introducción

Formularia es una aplicación que pretende cubrir un campo que está poco explotado en los dispositivos móviles: la posibilidad de rellenar de forma correcta formularios con una serie de restricciones en cada campo. Estas restricciones sirven para que el programador del formulario-modelo pueda guiar al usuario a la hora de rellenar un formulario. Si el programador decide que en el campo *nombre* del formulario solo puede venir un nombre lo indicará en la sección de restricciones del campo, así cuando el usuario rellene el formulario se le indicará que en ese campo sólo se puede introducir un nombre válido (nombre compuesto por más de dos letras y sin números).

Figure 3.1: Formularia - Es una aplicación para sistemas Android 2.2 o superior.

Lo anteriormente dicho, junto con la posibilidad de rellenar estos formularios desde un dispositivo móvil Android, son dos de las tres finalidades de la aplicación Formularia. La tercera finalidad de esta aplicación es explicar y dar la infraestructura necesaria para poder almacenar estos formularios, tanto el modelo como el formulario-usuario, los usuarios, los grupos y los permisos necesarios para poder hacer uso de la aplicación de una forma correcta.

3.2 La aplicación

3.2.1 Vistas

Formularia dispone de distintas vistas para poder realizar todos las tareas necesarias de la aplicación: autenticarse, crear o abrir un nuevo formulario nuevo, visualizar un formulario ya rellenado o poder cambiar las preferencias de la aplicación, entre otras vistas. A continuación se describen en profundidad cada una de ellas.

3.2.1.1 Login

En esta vista se muestran varias opciones. La principal es la de iniciar sesión con un nombre de usuario y una contraseña y la opción de poder iniciar sesión automáticamente a la hora de iniciar la aplicación. Si el usuario o la contraseña son erróneas o no se está conectado a Internet a la hora de pulsar en el botón de login aparecerá un mensaje avisando del error.

Figure 3.2: Vista Login - Vista inicial de la pantalla y el aviso de error al introducir un usuario erróneo.

Desde esta vista se puede acceder, pulsando en el botón de menú, a las opciones de menú que son dos:

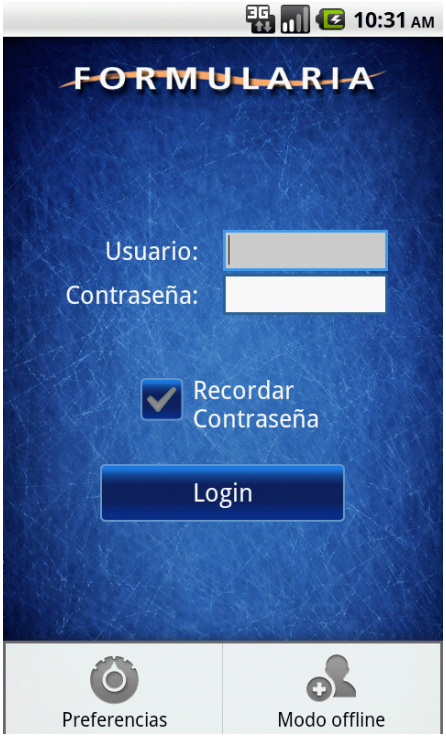

Figure 3.3: Vista Login - Opciones disponibles al pulsar sobre el botón de menú del móvil.

- *Modo sin conexión:* desde esta opción se puede entrar en la aplicación sin necesidad de usuario o contraseña válidos o si no se tiene conexión a Internet, pero sólo se puede acceder a los modelo-formularios previamente dispuestos para ello desde un usuario registrado.

- *Preferencias:* desde esta opción se puede configurar la url y el puerto del servidor externo donde se encuentra la base de datos externa y la opción de poder conectarse mediante el uso de la conexión de datos, además de wifi, del dispositivo móvil.

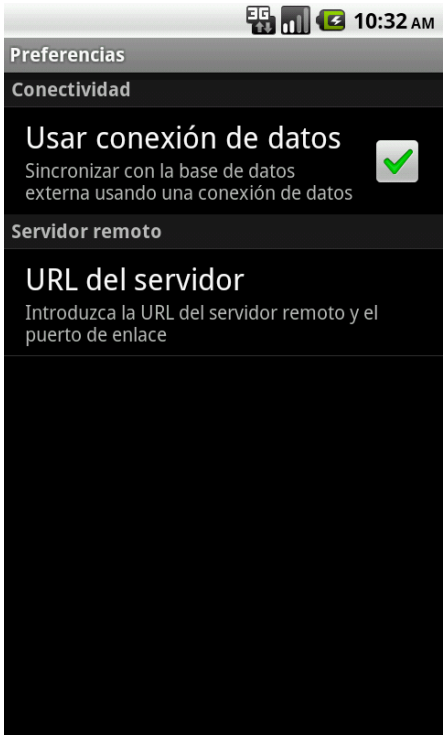

Figure 3.4: Vista Login - Opciones accesibles en el menú de preferencias desde la vista de Login.

3.2.1.2 Pantalla Principal

En esta pantalla hay dos botones:

- *Nuevo:* desde este botón se accede a otra vista donde se puede seleccionar el formulario-modelo desde el que se desea crear un nuevo formulario.
- *Abrir:* al pulsar este botón se accede a los formularios-usuarios disponibles para ese usuario.

Si se pulsa el botón de menú del móvil se accede a tres opciones. A estas tres opciones se podrá acceder desde cualquier otra vista durante toda la aplicación excepto la de Login.

Figure 3.5: Pantalla Principal - Vista general de la aplicación una vez logueado. Opciones disponibles en el menú.

- *Preferencias:* se explicará más adelante este menú.
- *Logout:* para poder salir a la vista de Login para cambiar de usuario.

- *Sincronizar:* al activar esta opción los formularios-usuarios del usuario actual se sincronizarán automáticamente con el servidor externo. Esta opción tiene varias preferencias seleccionables desde el menú de preferencias.

3.2.1.3 Nuevo Formulario

En esta vista aparece una lista con los formularios-modelos accesibles desde el usuario actualmente logueado en la aplicación. En esta lista cada formulario-modelo aparece con un nombre del formulario, una descripción y un icono. Este icono indica si el formulario-modelo está en el servidor remoto sólo o también está guardado en el dispositivo. Si lo tenemos almacenado en el dispositivo, se puede acceder a este formulario-modelo desde el modo sin conexión.

Figure 3.6: Nuevo Formulario - Lista de formularios-modelos disponibles para el usuario actual.

Al pinchar sobre un formulario-modelo aparecerá una ventana nueva en la que se tendrá que introducir el nombre con el que se quiere identificar el formulario-modelo que se va a rellenar.

Al mantener pulsado sobre un formulario-modelo aparecerá un menú contextual que nos permite guardar el formulario en la base de datos interna del dispositivo.

Figure 3.7: Nuevo Formulario - Diferentes opciones del menú contextual al mantener pulsado sobre un formulario-modelo.

3.2.1.4 Abrir Formulario

Desde esta vista se puede acceder a una lista con los formularios-usuarios. Cada formulario-usuario tiene el nombre con el que se guardó, la descripción del formulario-modelo del que procede y el nombre del usuario al que pertenece (esto sirve para identificar si hay algún formulario-usuario rellenado en el modo sin conexión y, así, poder reclamarlo para el usuario actual). Además de estos tres campos cada formulario tiene un icono diferente para indicar el estado de sincronismo entre la base de datos interna y la externa. El significado de estos iconos se explica en el punto 3.2.4 de esta memoria.

Al pulsar sobre un formulario se pasa a la Vista Previa del mismo donde posteriormente se podrá editar si así se desea.

Al mantener pulsado sobre el formulario rellenado por el usuario aparecerá un menú contextual que nos permitirá aplicar las siguientes opciones:

Figure 3.8: Abrir Formulario - Lista de formularios-usuario disponibles para el usuario actual.

- *Guardar como copia:* permite guardar una copia del formulario-usuario seleccionado.
- *Eliminar:* permite eliminar el formulario-usuario seleccionado.
- *Sincronizar:* permite sincronizar el formulario-usuario seleccionado. Esto depende de las opciones activadas en el menú de preferencias.

Si el formulario-usuario proviene del modo sin conexión sólo se dispondrá de la opción *Reclamar formulario*. Con esta opción se puede reclamar un formulario-usuario rellenado en el modo sin conexión y, así, este formulario pasará a pertenecer al usuario actual.

3.2.1.5 Vista Previa

Vista Previa tiene diferentes visualizaciones dependiendo de qué vista anterior se proceda o de si el formulario es correcto o no.

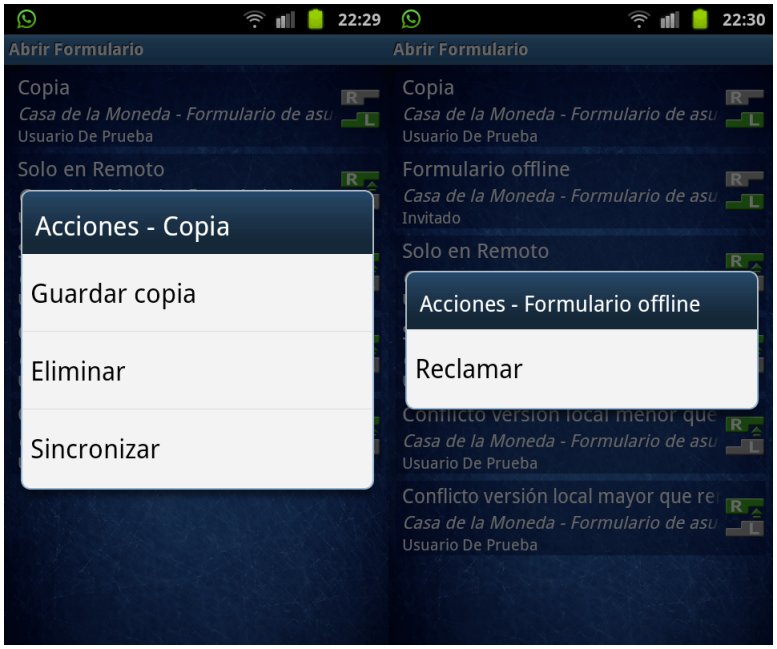

Figure 3.9: Abrir Formulario - Menú contextual que se despliega al mantener pulsado sobre un formulario-usuario.

Si se procede de Nuevo Formulario la Vista Previa tendrá dos elementos. Una previsualización del formulario-modelo sin rellenar y un botón *Editar*, que al pulsar sobre él pasaremos a la vista de Rellenar Formulario.

Si se procede de Rellenar Formulario sólo se verá la visualización del formulario-modelo con los campos que estén rellenos hasta el momento.

Si se procede de Abrir Formulario o Rellenar Formulario y el formulario está sin completar, aparecerá la visualización del formulario-usuario y el botón de *Editar*. Si el formulario está completo y bien relleno aparecen las opciones de *Compartir* y *Editar*. Con Compartir se puede enviar un .jpg de la Vista Previa formulario rellenado por email u otras aplicaciones que hagan uso del *Intent Extra_Stream* de Android.

3.2.1.6 Rellenar Formulario

En esta ventana nos aparecerán todos los elementos necesarios para rellenar el formulario-modelo de forma adecuada. Esta pantalla depende de cómo esté programado el formulario-modelo en el XML. Normalmente aparecerá un título y una o varias casillas para in-

Figure 3.10: Vista Previa - Previsualización del formulario-modelo.

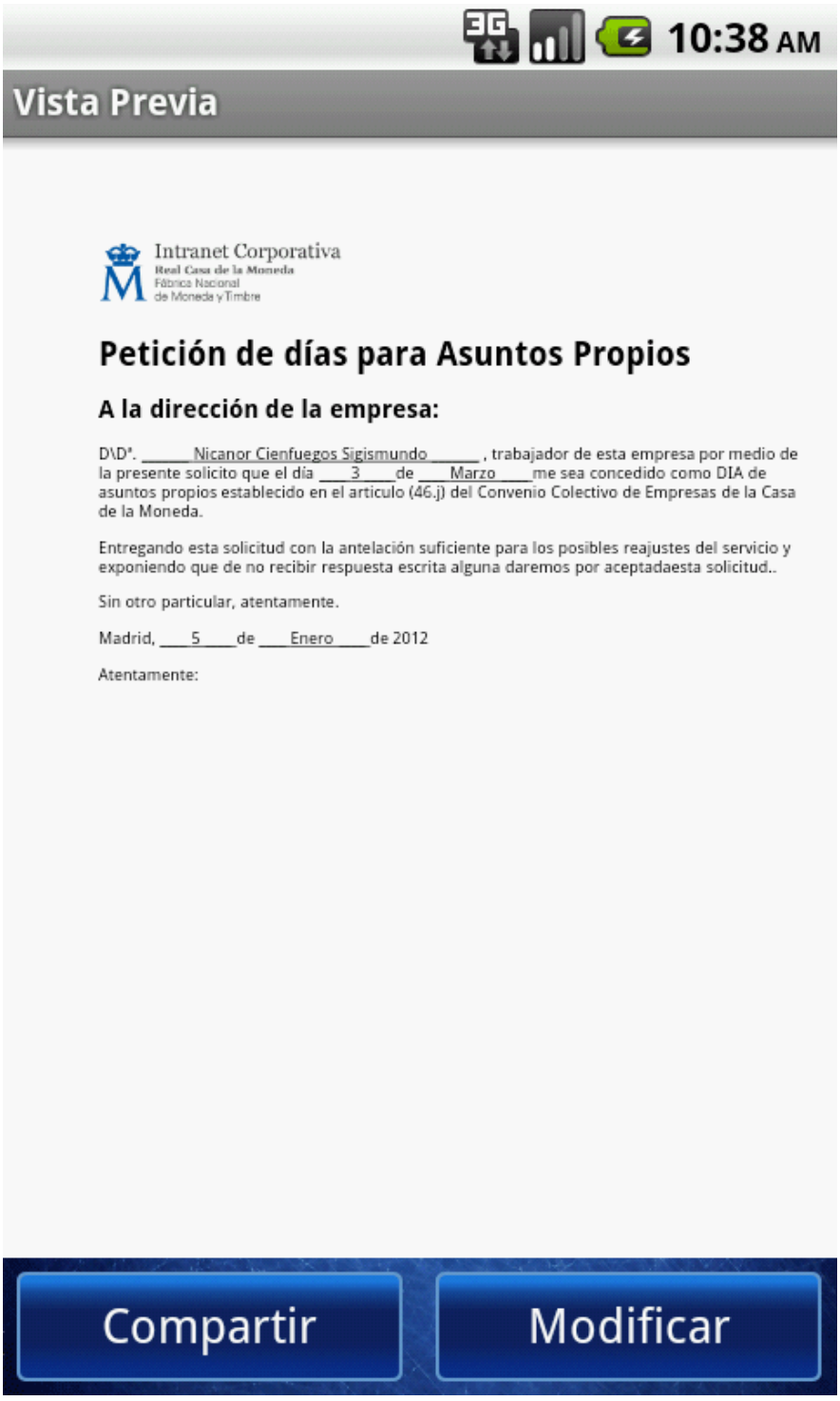

Figure 3.11: Vista Previa - Vista previa del Formulario-usuario rellenado.

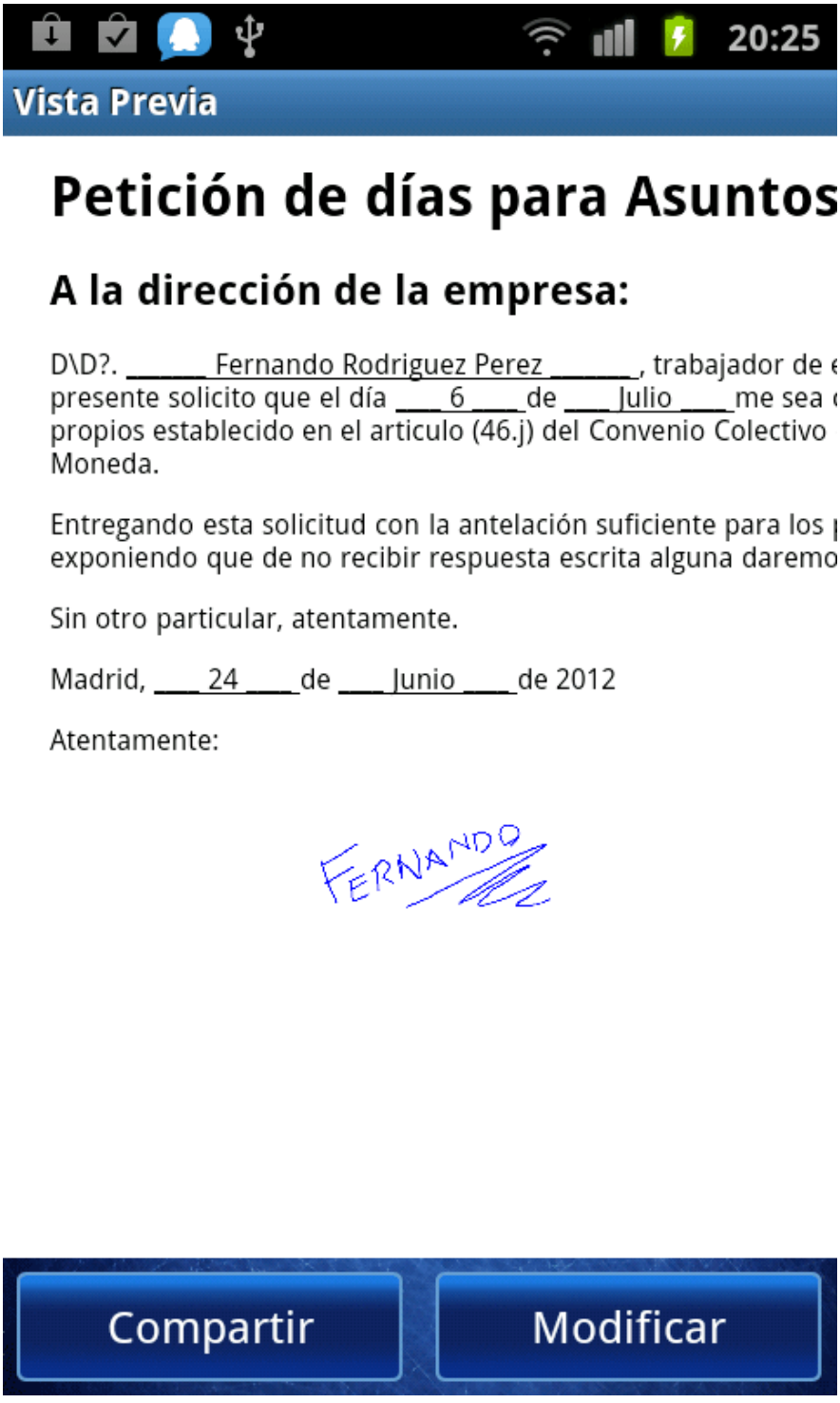

Figure 3.12: Vista Previa - Formulario con una firma realizada por el usuario.

troducir los datos necesarios para la correcta cumplimentación del mismo. Si la casilla está correctamente rellenada se coloreará el borde de verde, si no lo está el borde será rojo y saldrá un aviso al desmarcar la casilla con un mensaje explicativo indicando qué hay que introducir para que sea correcto.

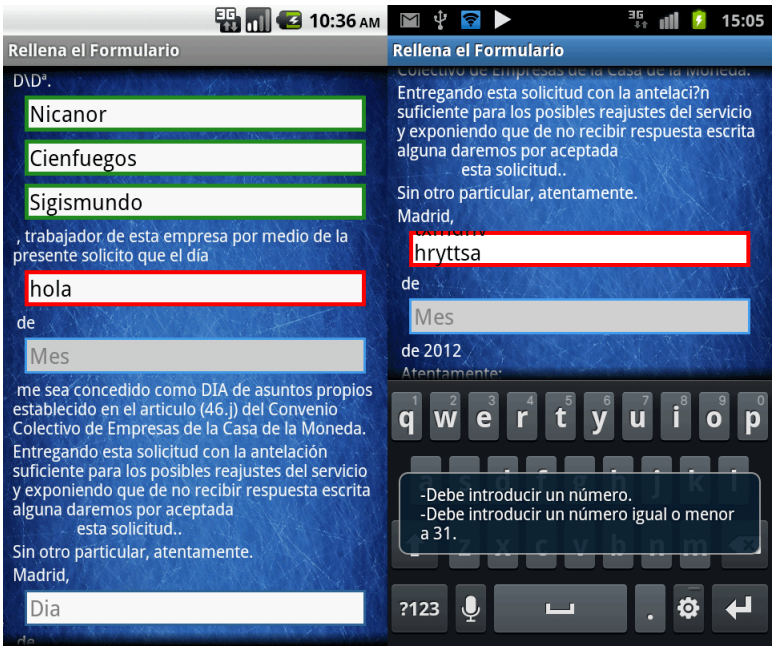

Figure 3.13: Rellenar Formulario - Campos rellenados correctamente aparecen en verde, los incorrectos en rojo. Aviso de como rellenar el campo en rojo de forma correcta.

Adicionalmente podrán encontrarse elementos como casillas de verificación y cuadros de firmas. Las casillas de verificación podrán marcarse o desmarcarse. Al pulsar un cuadro de firmas, la aplicación pasará a la interfaz de firmar.

Al final de la ventana aparecerán dos botones:

- *Vista Previa:* lleva al usuario a la pantalla de Vista Previa.

- *Guardar:* al pulsar sobre este botón aparecerá una ventana emergente preguntando dónde se quiere guardar el formulario: en el servidor externo o en el dispositivo móvil. Una vez guardados, los formularios-modelos pasan a ser formularios-usuarios y sólo podrán ser accedidos por ese usuario.

Figure 3.14: Firmar - Visualización de como quedaría la firma en la vista Rellenar Formulario.

3.2.1.7 Firmar

Esta ventana es muy sencilla. Se compone simplemente de un fondo negro sobre el que se pueden realizar trazos. Una vez realizada la firma, una pulsación en el botón *Atrás* retornará a la vista Rellenar Formulario actualizando la imagen del cuadro de firmas.

3.2.1.8 Preferencias

Desde este menú se pueden seleccionar diversas preferencias:

- *Usar conexión de datos:* permite sincronizar con la base de datos externa usando una conexión de datos. Si el usuario no quiere consumir conexión de datos es recomendable desactivar esta opción y usar la conexión wifi del dispositivo.
- *Servidor externo:* al activar esta opción tienen prioridad las copias de los formularios guardadas en el servidor externo. De esta forma los formularios del dispositivo se sobreescribirán sin preguntar al usuario si se encuentra una versión más reciente.

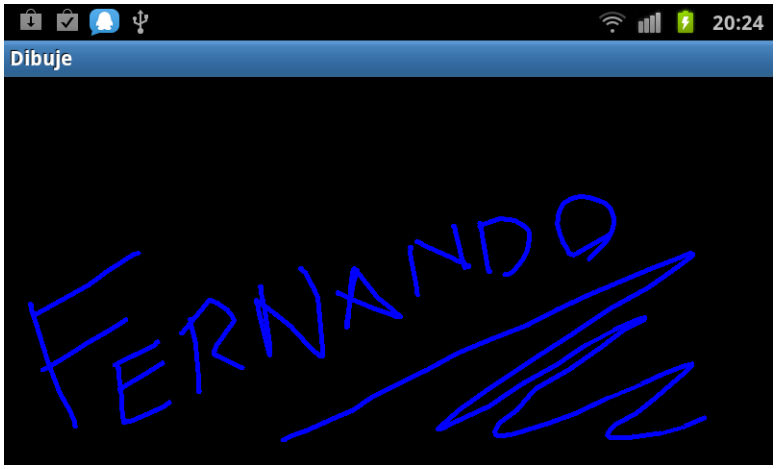

Figure 3.15: Firmar - Pantalla para introducir la firma.

- *Sincronizar locales:* cuando se activa esta opción los formularios locales se suben al servidor automáticamente en la sincronización.
- *No mantener datos:* esta preferencia, cuando está activa, no mantiene ningún dato en el dispositivo al sólo trabajar con los datos del servidor remoto. Debido a esto no se podrán almacenar formularios sin conexión y no se podrá entrar en modo sin conexión.
- *Servidor remoto:* permite escribir la dirección del servidor remoto y el puerto del mismo.

Para la vista de Login el menú de preferencias sólo tiene la opción de *Usar conexión de datos* y *Servidor remoto*.

3.2.2 Edición de formularios

La edición de formularios es la función principal de la aplicación Formularia. Con ella se puede rellenar de una forma correcta (según esté programado el formulario en el XML) un formulario-modelo. Un formulario se compone de campos de edición, etiquetas de texto y cuadros de marcado.

- *Etiquetas de texto:* es donde aparece el texto fijo de un formulario. "Nombre y Apellidos" sería un ejemplo, "Dirección" otro o el título del formulario sería otro.

Figure 3.16: Preferencias - Ventana de preferencias en la que se puede modificar, entre otras cosas, la prioridad de sincronización de los formularios.

- *Campos de edición:* es donde el usuario va a introducir los datos con los que desea rellenar el formulario. Estos campos, por defecto, tienen un texto indicativo que debe introducir el usuario. Si el usuario introduce un valor correcto (viene indicado en la restricciones del campo en el XML, se explica en el punto 4 de la memoria) el campo de edición correspondiente pasará a tener los bordes verdes. Si el valor es incorrecto, el campo tendrá los bordes rojos y aparecerá en la parte inferior de la aplicación una ventana indicando los valores necesarios a introducir para que el campo sea correcto (sólo admite números o debe ser una dirección de correo, por ejemplo).

- *Casillas de verificación:* son pequeñas casillas que tienen la posibilidad de estar marcadas o no. Estas casillas deben venir acompañadas de un texto indicativo junto a ellas.

- *Cuadros de firmas:* son imágenes que al pulsarlas, muestran una pantalla negra sobre la que se pueden realizar trazos como si de una pizarra se tratase. Una vez realizada la firma, el cuadro se actualiza mostrando una vista previa del trazo recién realizado.

Una vez abierta la ventana de edición de formulario se podrá guardar de dos maneras diferentes:

- *Sólo copia local:* se guarda el formulario-usuario en la base de datos interna del dispositivo.

- *Servidor:* el formulario-usuario se guarda tanto en la base de datos externa como en la interna del móvil.

3.2.3 Seguridad

Desde el punto de vista de la seguridad, cada usuario posee un nombre de usuario y una contraseña con la que puede acceder a la aplicación desde la pantalla de Login. Además, cada usuario tiene permitido el acceso a formularios modelo que vienen determinados

por los grupos a los que pertenece el usuario, que tienen a su vez permisos sobre los formularios, y los permisos asociados al usuario directamente. Estos permisos deben ser definidos por el administrador de la base de datos.

Cada usuario puede acceder a los formularios usuario que ha creado y, además, a los que han sido creados con el modo sin conexión para poder reclamarlos. Para crear estos últimos no hace falta tener conexión a Internet y se debe entrar como invitado, usando el modo sin conexión como se ha descrito en el apartado dedicado a la pantalla de Login.

3.2.4 Conectividad

La parte de conectividad de Formularia es uno de los servicios más importantes de la aplicación. Nos permite poder compartir los formularios con un servidor externo o con otras aplicaciones del dispositivo móvil (siempre bajo la responsabilidad del usuario).

La conectividad con otras aplicaciones del móvil se puede realizar desde la Vista Previa de la aplicación, haciendo uso del botón *Compartir* (sólo se puede hacer si el formulario está correctamente rellenado y se dispone de conexión a Internet). Una vez pulsado el botón, la aplicación realiza un captura de la visualización del formulario y ésta puede ser compartida con otras aplicaciones que se tengan en el dispositivo, que permitan recibir imágenes *.jpg*, por ejemplo: mail, *whatsapp, dropbox, ubuntuone...*

La conectividad con el servidor externo es uno de los puntos clave de esta aplicación: el poder mantener sincronizados distintos formularios, ya sean modelo o usuario, con distintos dispositivos móviles mediante un servidor, es lo que permite usar esta aplicación de una forma eficiente. Por ello, en la aplicación se pueden diferenciar distintos estados de los formularios con respecto a la base de datos externa e interna.

3.2.4.1 Formulario-modelo

El formulario-modelo sólo tiene dos estados:

- *Sólo servidor remoto:* el formulario-modelo está en la base de datos externa.

Figure 3.17: Sólo servidor remoto - Sólo podrás acceder a este formulario si tienes conexión a Internet.

- *Sincronizado:* el formulario-modelo está tanto en la base de datos externa como en la base de datos del dispositivo móvil. Esto permite poder rellenar formularios estando en el modo "sin conexión" o, en caso de haber logueado antes y haber perdido temporalmente la conexión, admite poder rellenar los formularios disponibles localmente.

Figure 3.18: Sincronizado - Para poder trabajar con un formulario-modelo en modo sin conexión necesitas tenerlo guardado localmente en el dispositivo.

3.2.4.2 Formulario-usuario

El formulario-usuario permite varios estados de sincronización:

- *Sólo servidor remoto:* el formulario-usuario está en la base de datos externa.

Figure 3.19: Sólo servidor remoto - El formulario-usuario sólo está almacenado en el servidor externo.

- *Sólo servidor local:* el formulario-usuario sólo es accesible desde el dispositivo móvil

Figure 3.20: Sólo servidor local - El formulario-usuario es una copia local.

- *Sincronizado*: el formulario-usuario está tanto en la base de datos externa como en la base de datos del dispositivo móvil. El formulario-usuario está completamente sincronizado en las dos bases de datos.

Figure 3.21: Sincronizado - El formulario-usuario está tanto en el servidor externo como en el móvil y son la misma versión.

- *Versión más nueva en el servidor remoto:* en la base de datos externa hay una versión más nueva del formulario-usuario y avisa de que haría falta sincronizar.

- *Versión más nueva en el servidor local:* en la base de datos interna hay una versión más nueva del formulario-usuario y avisa de que haría falta sincronizar para tener una copia actualizada en el servidor externo.

Figure 3.22: Versión más nueva en el servidor remoto - Hace falta actualizar el formulario del móvil.

Figure 3.23: Versión más nueva en el servidor local - Hace falta subir el formulario al servidor externo para actualizar la copia allí almacenada.

- *Sucio:* las versiones del formulario-usuario son iguales en el servidor externo y en el dispositivo, pero en este último ha sido modificado mientras no había una conexión activa, por lo que es necesaria una sincronización.

Figure 3.24: Sucio - Se necesita sincronizar para actualizar el formulario.

3.3 La base de datos

La base de datos es un componente esencial de la aplicación que se encarga de guardar los datos de los distintos usuarios, seguridad, formularios-modelo, formularios-usuario, etc. Debido al comportamiento de la aplicación nos encontramos con dos tipos de base de datos.

3.3.1 Base de datos interna

Esta es la base de datos que se encuentra localizada en el interior del dispositivo donde esté instalada la aplicación. Esta base de datos contiene los datos necesarios para una ejecución local de la aplicación (sin el apoyo del servidor) ya sea porque así se especifica o por la conectividad disponible en un momento dado. Estos datos incluyen los de los formularios-modelo que se quieran almacenar de forma local, así como los formularios-usuario que también se quieran almacenar de esta forma. Estos datos se sincronizarán o no con los de la base de datos externa según el usuario lo haya especificado.

Con la elección de diversas opciones en la aplicación esta base de datos puede llegar a ser innecesaria, haciendo así que toda la información sólo pueda ser obtenida de la base de datos externa.

3.3.2 Base de datos externa

Esta es la base de datos que se encuentra localizada en un dispositivo o servidor externo al dispositivo donde esté instalada la aplicación y es capaz de servir información a distintos dispositivos móviles con distintos usuarios. Para acceder a ésta es necesario contar con la conectividad necesaria.

Contiene la información relativa a la seguridad de acceso a los formularios, tanto en el nivel de grupo como de usuario. También contiene la información de grupos y los grupos a los que pertenece cada usuario, así como la información de usuarios. Además incluye la información de los formularios modelo que vayamos a utilizar en el editor y los formularios de usuario que han sido generados y sincronizados con esta base de datos externa.

En un punto posterior se tratará la estructura concreta de ambas bases de datos.

3.4 Programación de formularios

Es esencial para el uso de la aplicación la existencia de formularios-modelo creados por los administradores del servidor externo sobre el que esté funcionando la aplicación. Estos formularios son creados de la siguiente forma:

1. Se crea un archivo XML con el contenido de los elementos de dicho formulario.

2. Se crea el archivo XSLT que da forma a esos formularios.

A partir de estos dos archivos, la aplicación es capaz de generar un documento html bien formateado, el cual se muestra como resultado.

Estos archivos han de ser proporcionados por un usuario administrador y han de ser incluidos en la base de datos para su posterior disponibilidad para los usuarios.

4

Desarrollo e Implementación

4.1 Introducción

Formularia está construido sobre Android 2.2 mediante Eclipse en el lenguaje Java. Por otra parte está construido el servidor externo mediante Apache Tomcat en Netbeans con el lenguaje Java, el cual puede ser reemplazado por cualquier otro que acepte y devuelva las peticiones de la misma forma que lo hace el implementado.

En esta sección se describe el desarrollo seguido para la construcción de la aplicación, así como cada una de las partes que la componen.

4.2 Método de Desarrollo

Al inicio del proyecto se decidió dividir este en módulos o partes. La realización de cada uno de estos módulos fue encargada a una persona diferente bajo unos requerimientos definidos, para luego proseguir con su integración en el proyecto. De este modo se sigue una estructura descentralizada en la que las decisiones de cada parte del proyecto, mientras no afecten a la totalidad o a otra parte, las decide la persona encargada de estas. Los módulos en los que se decidió dividir el proyecto fueron: interfaz, editor y bases de datos.

La parte de interfaz comprende la estética de la aplicación e interacción con esta. En este módulo se incluyen la realización de las distintas vistas, excepto la de edición,

con la preparación de las distintas estructuras que deben mostrar los datos a partir de los obtenidos de la base de datos.

El módulo de edición se encarga de construir y mostrar la vista de edición de formulario a partir de un XML obtenido de la base de datos, y la vista previa a partir de un XSLT asociado. También es la parte encargada de actualizar el contenido del XML una vez se ha terminado de editar un formulario, que es el paso previo a guardarlo en la base de datos.

Por último, el módulo de la base de datos se encarga de gestionar, tanto la base de datos interna del dispositivo, como la base de datos externa, para servir los datos necesarios.

La primera dificultad de este proyecto residía en que ninguno de los integrantes del grupo había trabajado en el desarrollo de una aplicación en Android ni en ninguna otra plataforma móvil. Por ello llevó un tiempo acostumbrarse a las particularidades de un lenguaje que, aunque utiliza funciones y procedimientos de Java, tiene sus propios métodos y variables. Además de esto las bases de datos utilizadas limitaban el tipo de datos a utilizar, no teniendo SQLite tipo de datos fecha por ejemplo, lo que hizo que se tuvieran que replantear algunos conceptos.

Una vez terminado un módulo, este se ponía en común y se analizaba si el diseño era correcto, faltaban funcionalidades o había que implementar otras nuevas. Terminado este proceso con todos los módulos, se pasaron a integrar todos en el proyecto. Una vez terminada esa parte del desarrollo, pasamos a hacer iteraciones o etapas de trabajo en las que se analizaba si se cumplían las especificaciones acordadas y, en caso de que estas no se cumplieran, se pasaba de nuevo a la fase de implementación para hacer las correcciones especificadas.

Algunas de las características que más iteraciones necesitaron fueron la seguridad, la interacción con la base de datos y la edición de los formularios. En el primer caso, mientras se avanzaba en el desarrollo, se vió que la seguridad planteada no era suficiente, por lo que se tuvo que desarrollar un modo sin conexión ya que no se podía guardar en la base de datos interna ninguna credencial. Este modo sin conexión, a su vez, hizo que la base de datos sufriera cambios que provocaron que el número de iteraciones

necesarias fuera superior, ya que había que asegurar que los datos eran suficientes en el modo sin conexión. Además para la base de datos externa, en un estadío temprano del desarrollo se decidió que sería implementada en el propio dispositivo, para facilitar el desarrollo y control de errores, por lo que posteriormente tuvo que trasladarse a un servidor externo que también tuvo que ser implementado. En cuanto a la edición de formularios, esta fase tuvo más iteraciones debido a que se decidió implementar diversas características y funcionalidades adicionales, que no se habían tenido en cuenta en un principio, pero que se consiedaron necesarias posteriormente.

Una vez terminado todo esto se pasó a la etapa de prueba de la aplicación. En esta, tratamos de simular un uso normal de la aplicación para localizar los últimos errores y corregirlos antes del lanzamiento de una versión estable.

Respecto a la arquitectura de la aplicación se ha seguido el patrón de diseño MVC (Modelo, Vista, Controlador), un patrón de diseño fundamentado en encapsular por separado el almacenamiento de los datos (Modelo), la interfaz que muestra esos datos (Vista), y la lógica de negocio (Controlador).

Este patrón se aplica, en una forma general, para toda la aplicación en la que el modelo correspondería con las bases de datos, las vistas con las distintas interfaces y el controlador con las clases que gestionan la comunicación con la base de datos.

Además este patrón se aplica también de forma particular a la interfaz de edición de formularios, en cuyo caso la vista corresponde con la interfaz, y el modelo (que contiene los datos del formulario que se está editando en el momento en el que se muestra la vista) está mantenido en la memoria del terminal telefónico móvil.

Sin embargo, debido a que la interfaz de edición de formularios ha de ser generada dinámicamente y de forma dependiente de los datos contenidos en el modelo, el controlador no realiza todas las funcionalidades que debería. Esto es debido a que en el desarrollo de una aplicación para el sistema operativo android, cada vista que se muestra al usuario es generada en una nueva actividad. Y es esta actividad la que carga con la labor lógica de generar los contenidos de la interfaz.

Además el modelo está estructurado de tal forma que cada uno de sus elementos dispone de los datos necesarios para comprobar su validez. Dado que tanto la fun-

cionalidad de generar la vista es soportada por ésta misma y que la funcionalidad de mantener la estabilidad estructural de los datos contenidos en el modelo es soportado por éste, el controlador realiza más una labor de comunicación entre vista y modelo que de control y computación.

4.3 Estructura de la base de datos

La estructura de la base de datos puede dividirse en dos partes diferenciadas. Por un lado tenemos la parte de la seguridad que solo se encuentra en el servidor externo y por el otro, tenemos la parte de almacenamiento de formularios donde se encuentran los formularios modelo y los formularios de usuario.

4.3.1 Base de datos interna

La base de datos interna solo puede contener datos sobre los formularios que el usuario guarda y los modelos que quiere tener el usuario en el modo local o sin conexión. Esto es así como una medida de seguridad más contra el acceso no deseado. Si en la aplicación el usuario decide no guardar datos en el dispositivo, esta base de datos estará vacía.

La tecnología utilizada para la implementación de esta base de datos es la de SQLite la cual se incluye nativamente en Android. Este sistema tiene múltiples limitaciones que se reflejan en las distintas particularidades de la construcción de la base de datos. Por mencionar algún ejemplo el tipo de datos para fechas no existe por lo que usamos un entero con los milisegundos representando el tiempo UNIX.

La estructura es la siguiente :

FORMULARIO_MODELO:

- *ID_FORMULARIO:* un autonumérico para que cada formulario insertado pueda ser identificado independientemente.
- *ID_USUARIO:* identificador que representa al usuario propietario del formulario-modelo.
- *NOMBRE:* nombre del formulario.
- *DESCRIPCION:* descripción del cometido del formulario u otra información sobre este.
- *ARCHIVO_XML:* archivo XML que representa el formulario en forma de cadena.
- *ARCHIVO_XSLT:* archivo XSLT que representa la visualización del formulario en forma de cadena.
- *FECHA_CREACION:* fecha de creación del formulario.
- *FECHA_BAJA:* fecha de baja del formulario. Si es NULL significa que el formulario está activo.

El *ID_USUARIO* es necesario para saber qué usuario es el que ha hecho que el formulario-modelo representado por el registro esté disponible de forma local.

FORMULARIO_USUARIO:

- *ID_FORMULARIO_USUARIO_INTERNO:* un autonumérico para que cada formulario-usuario insertado pueda ser identificado independientemente.
- *ID_FORMULARIO_USUARIO:* identificador que representa el formulario-usuario en la base de datos externa. Si el formulario es solo local el valor de este campo será 0.

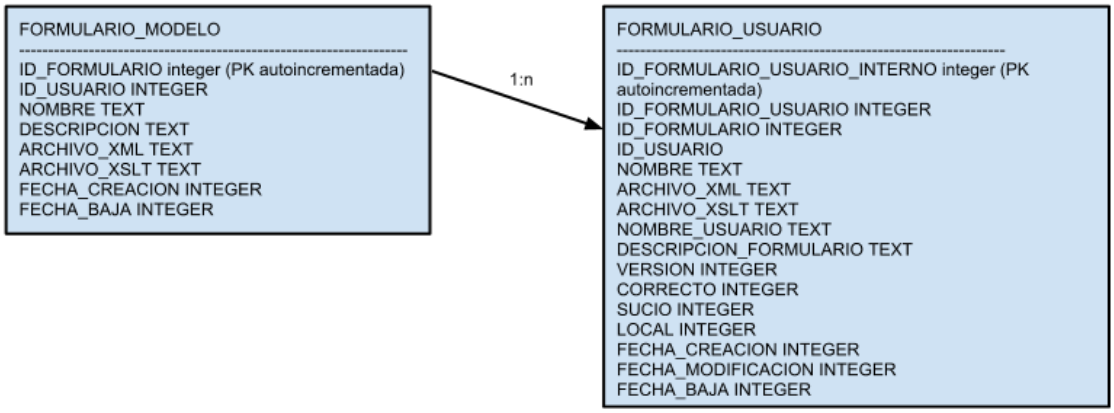

Figure 4.1: Base de datos interna - Esquema de como está implementada la base de datos interna.

- *ID_FORMULARIO:* identificador que representa el formulario-modelo del cual es instancia este formulario-usuario.
- *ID_USUARIO:* identificador que representa al usuario propietario del formulario-usuario.
- *NOMBRE:* Nombre dado a este formulario-usuario.
- *ARCHIVO_XML:* archivo XML que representa el formulario en forma de cadena.
- *ARCHIVO_XSLT:* archivo XSLT que representa la visualización del formulario en forma de cadena.
- *NOMBRE_USUARIO:* nombre del usuario propietario de este formulario-usuario.
- *DESCRIPCION_FORMULARIO:* descripción del modelo del que proviene este formulario-usuario.
- *VERSION:* número que representa la versión del formulario.

- *CORRECTO:* número que adquiere los valores 0(false) y 1(true) representando si el formulario está cumplimentado correctamente o no.
- *SUCIO:* número que adquiere los valores 0(false) y 1(true) representando si el formulario está cumplimentado correctamente o no.
- *LOCAL:* número que adquiere los valores 0(false) y 1(true) representando si el formulario solo se encuentra en la base de datos interna.
- *FECHA_CREACION:* fecha de creación del formulario.
- *FECHA_MODIFICACION:* última fecha de modificación del formulario.
- *FECHA_BAJA:* fecha de baja del formulario. Si es NULL significa que el formulario está activo.

En el caso de los formularios que solo son locales el campo *LOCAL* contendrá el valor 1 y el campo versión y el campo *ID_FORMULARIO_USUARIO* contendrán el valor 0. Además el campo *SUCIO* contendrá siempre un 1 en este caso. Los campos *NOMBRE_USUARIO* y *DESCRIPCION_FORMULARIO* son necesarios ya que en el caso de estar en el modo desconectado o sin conexión, estos campos no podrán ser obtenidos de otra forma que contenerlos en esta tabla.

4.3.2 Base de datos externa

La base de datos externa incluye la seguridad de la aplicación además de almacenar los formularios. A la hora de intentar acceder a un formulario hay dos niveles de seguridad: el nivel de usuario y el nivel de grupo. Si un usuario pertenece a un grupo y este tiene permitido el acceso a un modelo de formulario, el usuario podrá acceder al formulario. De la misma manera si el grupo tiene el acceso restringido lo tendrá también el usuario. Si un usuario tiene acceso permitido (o no permitido) a un modelo, directamente este permiso se superpone a los que el usuario pueda tener por pertenecer a un grupo, es decir, si está permitido por usuario y denegado por grupo podrá acceder de todas maneras.

La estructura es la siguiente:

FORMULARIO_MODELO:

- *ID_FORMULARIO:* un autonumérico para que cada formulario insertado pueda ser identificado independientemente.
- *NOMBRE:* nombre del formulario.
- *DESCRIPCION:* descripción del cometido del formulario u otra información sobre este.
- *ARCHIVO_XML:* archivo XML que representa el formulario en forma de cadena.
- *ARCHIVO_XSLT:* archivo XSLT que representa la visualización del formulario en forma de cadena.
- *FECHA_CREACION:* fecha de creación del formulario.
- *FECHA_BAJA:* fecha de baja del formulario. Si es NULL significa que el formulario está activo.

FORMULARIO_USUARIO:

- *ID_FORMULARIO_USUARIO:* un autonumérico para que cada formulario-usuario insertado pueda ser identificado independientemente.
- *ID_FORMULARIO:* identificador que representa el formulario-modelo del cual es instancia este formulario-usuario.
- *ID_USUARIO:* identificador que representa al usuario propietario del formulario-usuario.
- *NOMBRE:* Nombre dado a este formulario-usuario.
- *ARCHIVO_XML:* archivo XML que representa el formulario en forma de cadena.

- *ARCHIVO_XSLT:* archivo XSLT que representa la visualización del formulario en forma de cadena.
- *VERSION:* número que representa la versión del formulario.
- *CORRECTO:* número que adquiere los valores 0(false) y 1(true) representando si el formulario está cumplimentado correctamente o no.
- *FECHA_CREACION:* fecha de creación del formulario.
- *FECHA_MODIFICACION:* última fecha de modificación del formulario.
- *FECHA_BAJA:* fecha de baja del formulario. Si es NULL significa que el formulario está activo.

USUARIOS:

- *ID_USUARIO:* un autonumérico para que cada usuario insertado pueda ser identificado independientemente.
- *APELLIDO1:* primer apellido del usuario.
- *APELLIDO2:* segundo apellido del usuario.
- *NOMBRE:* nombre del usuario.
- *USUARIO:* usuario para poder autenticarse en la aplicación.
- *EMAIL:* dirección email del usuario (reservado para un uso futuro).
- *PASS_HASH:* hash de la contraseña de autenticación.

GRUPOS:

- *ID_GRUPO:* in autonumérico para que cada grupo insertado pueda ser identificado independientemente.
- *DESCRIPCION:* descripción del grupo.

USUARIO_GRUPO:

- *ID_USUARIO_GRUPO:* un autonumérico para que cada relación de usuario y grupo insertado pueda ser identificado independientemente.
- *ID_USUARIO:* identificador que representa al usuario que se va a relacionar con el grupo.
- *ID_GRUPO:* identificador que representa al grupo en el que se va a incluir al usuario.

FORMULARIO_SEGURIDAD_USUARIO:

- *ID_USUARIO_FORMULARIO:* un autonumérico para que cada relación de usuario y seguridad insertada pueda ser identificado independientemente.
- *ID_USUARIO:* identificador que representa al usuario que se va a relacionar con un formulario y un permiso.
- *ID_FORMULARIO:* identificador que representa al formulario que se va a relacionar con un usuario y un permiso.
- *PERMISO:* cadena que representa el tipo de permiso. "DENEGADO" si se quiere denegar el acceso y "PERMITIDO" si representa que está permitido el acceso. Este campo no es binario para permitir futuras configuraciones.

FORMULARIO_SEGURIDAD_GRUPO:

- *ID_GRUPO_FORMULARIO:* un autonumérico para que cada relación de grupo y seguridad insertada pueda ser identificado independientemente.
- *ID_GRUPO:* identificador que representa al grupo que se va a relacionar con un formulario y un permiso.
- *ID_FORMULARIO:* identificador que representa al formulario que se va a relacionar con un grupo y un permiso.
- *PERMISO:* cadena que representa el tipo de permiso. "DENEGADO" si se quiere denegar el acceso y "PERMITIDO" si representa que está permitido el acceso.

4.3.3 Posibles mejoras

Durante el desarrollo de la base de datos, ésta ha sufrido diversos cambios debido a cambios en la seguridad y otros motivos. Aún así, ha habido características que no han podido o no se han sabido implementar.

Una posible mejora sería cifrar toda la información contenida en la base de datos, al menos la interna, para mejorar la seguridad. Esto complicaría las búsquedas y las distintas operaciones por lo que no se ha tenido en cuenta basándose en el tiempo de desarrollo.

Otra posible mejora pasa por incluir varios archivos XSLT por formulario ya que estos incluyen el estilo final del documento, pudiendo así obtener un formulario con distintos aspectos que pueda elegir el usuario.

Estas y otras mejoras serían posibles con una mayor profundización en el desarrollo.

4.4 Estructura de la aplicación

4.4.1 Views

El paquete view de Formularia se compone de todas las clases que implementan las vistas de la aplicación. Estas vistas son clases que extienden a la clase *Activity*. Esto

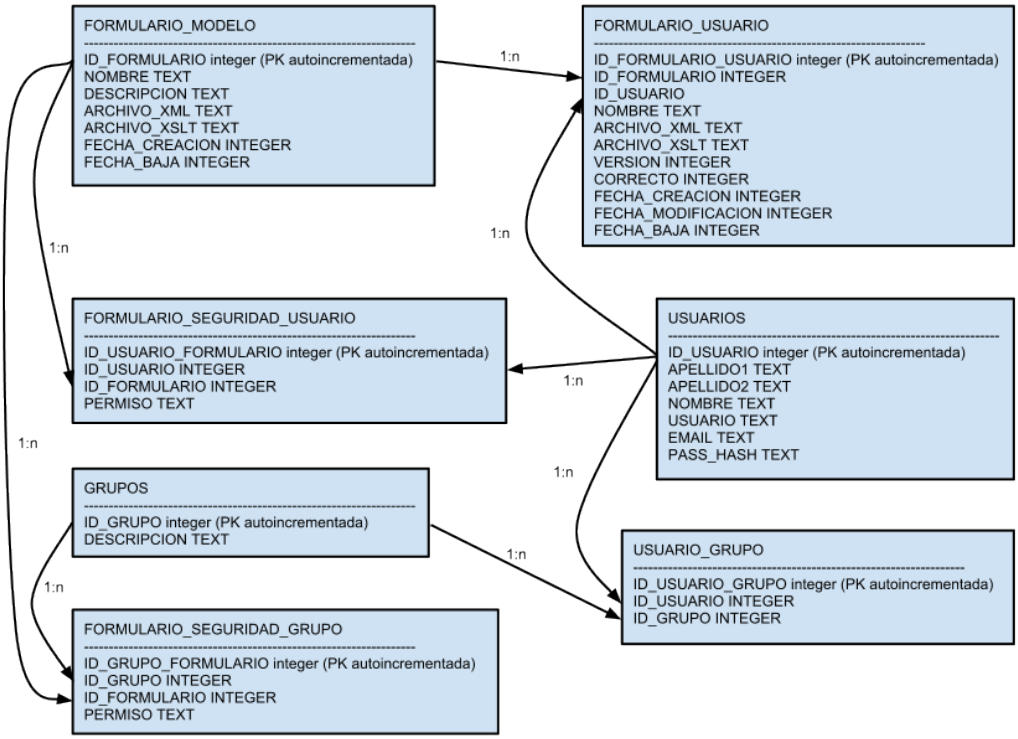

Figure 4.2: Base de datos externa - Esquema de como está implementada la base de datos externa.

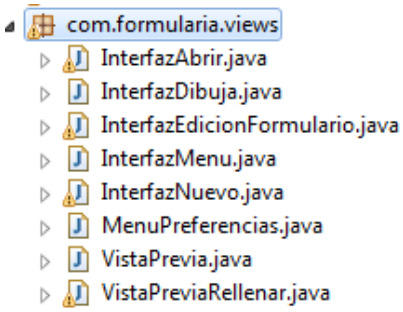

Figure 4.3: Paquete de las View - Interfaces de las que se compone el paquete.

permite “ejecutar” la clase en la máquina virtual de Android. Estas views o vistas hacen uso del método *setContentView* que permite lanzar la interfaz que se necesite.

```
public class InterfazMenu extends Activity {
    /** Called when the activity is first created. */
```

Figure 4.4: Declaración de una View - La clase InterfazMenu extiende Activity.

Las interfaces o las ventanas de la aplicación son el interfaz de la aplicación y se programan con XML en Android. Las interfaces son XML estrictos, y comienzan siempre con el encabezado `<?xml version="1.0" encoding="utf-8"?>` y todas las etiquetas abiertas deben ser debidamente cerradas. Dentro de cada interfaz (hay varios tipos) se insertan los componentes de la aplicación: botones, etiquetas, pulsadores etc. Cada uno de estos componentes se puede configurar con sus propiedades según sea necesario.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:background="@drawable/bc_fondo"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <ListView android:id="@+id/lstFormularios"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:scrollingCache="true"
        android:padding="6dp"
        android:dividerHeight="6dp"
        android:divider="#00000000"
        android:cacheColorHint="#00000000"
        />


</LinearLayout>
```

Figure 4.5: Trozo de código de un Layout - Ejemplo de cómo configurar una interfaz. El ListView es un componente que se puede configurar con sus propias características.

Cuando se quiere pasar de la pantalla de Login a la del Menú lo que se hace es lanzar una nueva Activity y está será la que contenga la view del Menú. Esto conlleva el poder

controlar la “vuelta atrás” de cada vista. Este proceso se debe controlar manualmente si se quiere que desde la view Menú no vuelva a la pantalla Login.

```
public void onBackPressed() {
    moveTaskToBack(true);
}
```

Figure 4.6: Código para mandar la aplicación a segundo plano - Desde el Menú principal al pulsar el botón de “atrás” la aplicación se pone en segundo plano.

En la aplicación Formularia hemos desarrollado la interfaz para que sufra el mínimo impacto entre las distintas resoluciones con las que puede funcionar el sistema operativo Android. Estas pueden ir desde las 3,7 hasta las 10 pulgadas que puede tener una tableta. Como la aplicación está pensada para poder trabajar con formularios, el tamaño de la pantalla y sus diferentes resoluciones son un punto muy importante de la misma por lo que se le ha dado una gran importancia a la compatibilidad con diferentes pantallas.

Cada vista de la aplicación está pensada para poder abrir su propio menú de preferencias, que según se esté trabajando en modo sin conexión o con conexión, muestra diferentes preferencias. Estas preferencias van desde usar la conexión de datos hasta la prioridad de sincronización de los formularios con la base de datos externa.

En Formularia los layouts están divididos en varios paquetes:

- *Paquete Layout:* en este paquete van las vistas generales de la aplicación, es decir, en esta carpeta es donde se programan los xml de los interfaces. Desde este paquete se puede acceder al login.xml, menu.xml, listaformularios.xml, etc.

- *Paquete Menu:* desde este paquete se accede a las interfaces que se cargan al darle al botón del menú desde el móvil. Estos cargan los botones de *Preferencias*, *Sincronizar* o *Logout*. Hay diferentes versiones según se esté en la vista login o en el resto de vistas de la aplicación, también puede cambiar si se está en modo sin conexión. En este paquete también se guardan los menús contextuales de la aplicación, estos pueden ser accedidos desde Abrir o Nuevo al dejar pulsado

```java
public void onBackPressed() {
    launchPreviousActivity();
}

/**
 * Lanza la actividad InterfazNuevo
 */
private void launchPreviousActivity(){
    if (nuevo){//Pasamos a items para modificar
        Intent nuevo = new Intent(this, InterfazNuevo.class);

        nuevo.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);

        Bundle bundle = new Bundle();
        bundle.putString("idUsuario", idUsuario);
        bundle.putString("nombreUsuario", nombreUsuario);
        nuevo.putExtras( bundle );

        startActivity(nuevo);

    }else{ //Pasamos a items nuevos
        Intent abrir = new Intent(this, InterfazAbrir.class);

        abrir.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);

        Bundle bundle = new Bundle();
        bundle.putString("idUsuario", idUsuario);
        bundle.putString("nombreUsuario", nombreUsuario);
        abrir.putExtras( bundle );

        startActivity(abrir);

    }
}
```

Figure 4.7: Método para controlar el flujo de las Views - Código para manejar la vuelta atrás de la Activitys seleccionando en cada momento cual es la Activity a la que se quiere volver.

Figure 4.8: Vista Login - Muestra de cómo se vería el Login a distintas resoluciones. La primera imagen a 3.7" y la segunda a 4.65".

Figure 4.9: Preferencias - Ventana de preferencias accesible desde menú principal de la aplicación.

sobre uno de los formularios. Las opciones que despliegan pueden ser: borrar, sincronizar, reclamar, guardar como copia...

- *Paquete XML:* este paquete sirve para poder configurar el XML del menú de preferencias. Los XML que contiene la carpeta configuran el menú de preferencias y sus distintas opciones. Hay dos tipos:

  - *menu_preferencias_login:* es el menú de preferencias que se carga desde la vista de Login y tiene una versión reducida de las preferencias: usar conexión de datos y la opción de insertar la dirección del servidor externo.

  - *menu_preferencias:* es el menú de preferencias principal de la aplicación y contiene todas las opciones: usar conexión de datos, no mantener datos locales en el móvil, sincronizar siempre los formularios-usuarios locales, prioridad del servidor externo...

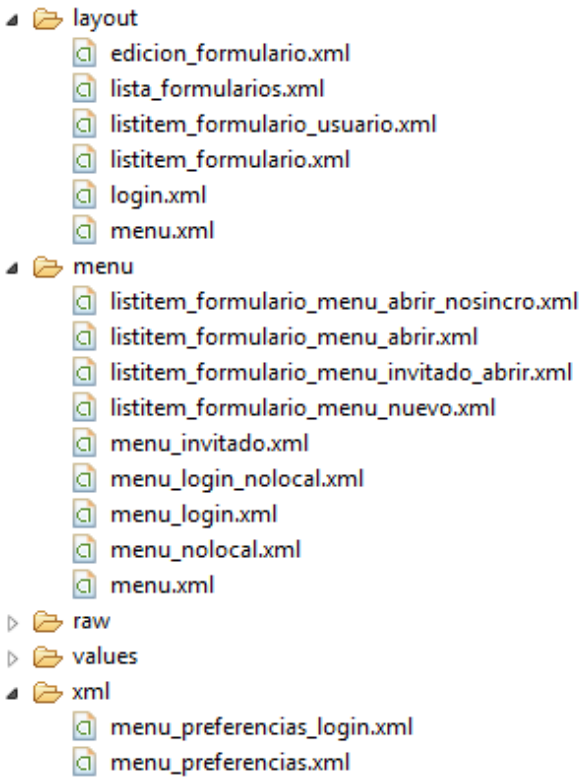

Figure 4.10: Layouts - Listado de los XML de los Layouts.

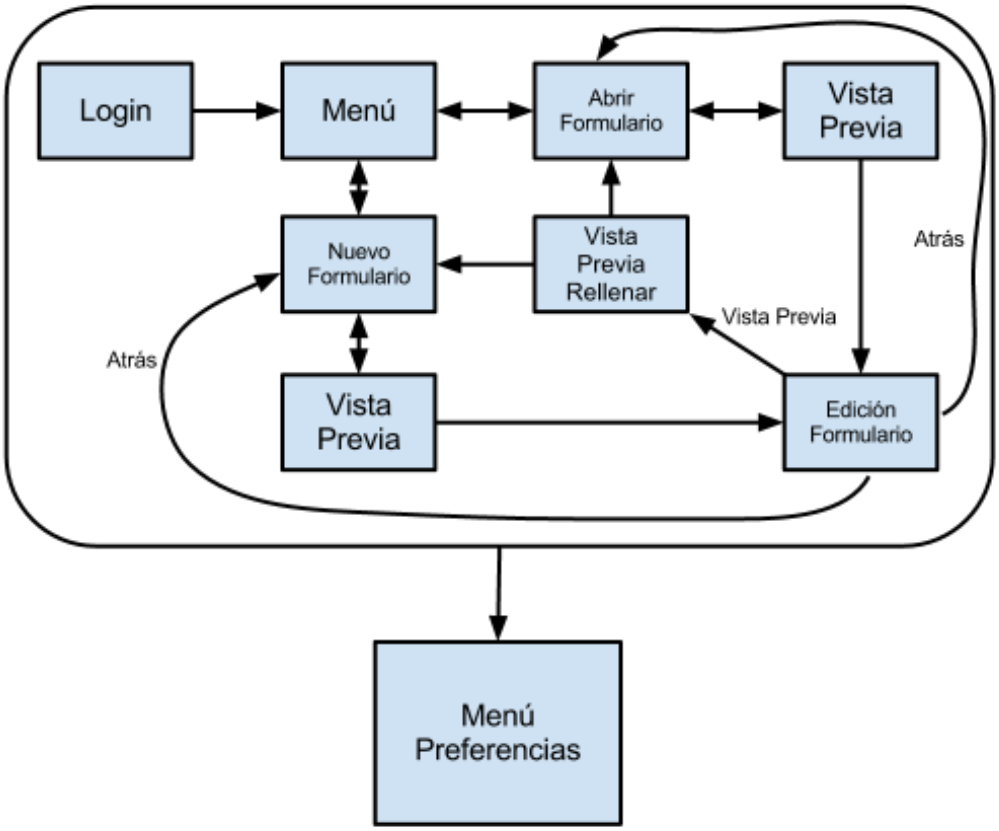

Figure 4.11: Views - Flujo de las Views.

4.4.2 Seguridad

Aunque en el apartado que explica la base de datos se introdujo la seguridad de acceso en este nivel, todavía queda tratar este tema en el nivel de aplicación. Los usuarios se encuentra en la base de datos externa con sus credenciales codificadas, por lo que es necesaria una conexión de datos activa para poder autenticar un intento de conexión. Es por esto por lo que si no hay una conexión con el servidor y el usuario intenta autenticarse, la aplicación no permitirá continuar, por lo que hay que utilizar el modo desconectado (o sin conexión) implementado.

En el caso de haber una conexión activa, el usuario y la contraseña se autentican contra la base de datos del servidor y si coincide se permite el acceso a la aplicación. Los usuarios no se guardan de ninguna forma en la base de datos interna por seguridad, y es por eso por lo que no se puede permitir una autenticación sin conexión. El modo sin conexión es un modo especial, en el cual sólo se pueden utilizar los formularios modelo guardados en el dispositivo; y los formularios creados con este modo no se pueden compartir de ninguna manera. Estos formularios son especiales y pueden ser reclamados por el usuario una vez se conecte a la aplicación autenticándose y si tiene permisos para editar formularios de ese tipo.

Otra medida de seguridad es no guardar ningún dato en el dispositivo. Para ello se ha incluido la opción “no mantener datos” en el menú de preferencias. Si se marca esta opción todos los datos guardados hasta ahora en la aplicación son borrados previa confirmación y a partir de ese momento no se guarda ninguno más hasta que no se desmarque. Esto por supuesto tiene una desventaja y es que no se pueden guardar formularios de forma local, ni modelos ni formularios de usuario, y tampoco se puede acceder al modo sin conexión.

4.4.3 Modelo

El modelo de la aplicación es el contenedor de los datos del formulario que se están tratando en este momento. En este caso se trata de un conjunto o lista de elementos

o “items” cada uno con sus debidos atributos o propiedades, entre ellas una lista de restricciones aplicables a dicho elemento.

El motivo para que estos elementos esten estructurados de esta forma es que sean capaces de validarse a sí mismos. Cada restricción tiene la capacidad de comprobar si es válida para el valor introducido en dicho elemento, de tal forma que cuando se modifica el valor de un elemento (como por ejemplo el texto introducido en un campo de escritura), comprobar si el valor introducido es correcto es tan sencillo como comprobar si cada una de sus restricciones es válida para dicho valor introducido.

De ser todo correcto, el elemento indica que la edición de su contenido se ha realizado satisfactoriamente. De no ser así, comunica cuáles de las restricciones han sido incumplidas y por qué, informando de ello al usuario.

4.4.3.1 Items

Dado que todos los elementos tienen cosas en común, todos heredan de una clase abstracta que contiene características genéricas para todos los elementos, como pueden ser los atributos id y restricciones, o la capacidad de ser editados, de preguntar su validez, o de recibir su conversión a formato xml.

Hasta el momento hay implementados los siguientes elementos:

- *Label:* se trata sencillamente de un texto. No es editable desde la interfaz de edición de formularios. Solamente tiene un atributo propio: el valor. Este es el texto que contiene, y que mostrará tanto en la pantalla de edición como en la vista previa, en caso de estar implementado en el XSLT.

- *EditText:* se trata de un campo de introducción de texto. Contiene 2 propiedades: un valor y una pista. La pista es un texto indicativo de lo que debe escribirse en dicho campo. El valor es el texto actualmente introducido en el campo de escritura. En caso de tener cualquier tipo de valor, no se mostrará la pista. Cuando el usuario modifica el contenido de un campo de escritura, éste automáticamente comprueba su validez con respecto a las restricciones con las que ha sido definido en el fichero XML. Todo lo referente a estas restricciones será explicado más

adelante. En caso de no resultar válido, mostrará una pequeña ventana emergente con las restricciones que han sido incumplidas y qué ha de hacerse para cumplirlas.

- *BoolBox:* se trata de una casilla de verificación. Contiene 2 propiedades: un texto de acompañamiento y su valor de marcado. El usuario puede, con total voluntad, marcar o desmarcar la casilla. El texto de acompañamiento se mostrará junto a la casilla.

- *Signature:* se trata de un elemento cuya finalidad es la de insertar una firma. Contiene una propiedad: el nombre del archivo que contiene la imagen de la firma. El usuario puede modificar la firma tantas veces como quiera.

4.4.3.2 Restricciones

Como ya se ha nombrado anteriormente, se pueden asociar restricciones a los elementos. Estas restricciones han de ser complementarias, pero nunca contrarias, pues han de cumplirse todas y cada una de ellas, y además por separado unas de otras.

Actualmente están implementadas las siguientes restricciones:

- *IsNumberRes:* el contenido del campo ha de ser un número.

- *IsIntegerRes:* el contenido ha de ser un número entero.

- *MaxValueRes:* indica el valor máximo de un número.

- *MinValueRes:* indica el valor mínimo de un número.

- *NoNumbersRes:* el campo no puede contener números.

- *MaxLengthRes:* longitud máxima en caracteres del contenido del campo.

- *MinLengthRes:* longitud mínima en caracteres del contenido del campo.

- *IsMonthRes:* el valor introducido debe equivaler a un mes.

- *IsDateRes:* el valor introducido debe equivaler a una fecha de la forma dd/mm/aaaa.

- *IsEmailRes:* el valor introducido debe equivaler a una dirección de correo electrónico válida.

4.4.3.3 Posibles mejoras

En caso de querer ampliar la aplicación se podrían añadir elementos adicionales como, por ejemplo, listas de selección desplegables o grupos de casillas de selección donde únicamente una puede estar marcada. Además, se podrían añadir infinidad de nuevas restricciones, siguiendo el patrón de que sean lo menos restrictivas posibles, para que puedan combinarse debidamente.

4.4.4 XMLHandler

Se trata de la unidad encargada de tomar por un lado el archivo XSLT y por el otro el XML, y fundirlos generando un archivo HTML. Para que esto sea posible, ambos archivos deben respetar los formatos que van a ser definidos y explicados posteriormente.

4.5 Formularios

Los formularios se definen mediante dos lenguajes: XML y XSLT.

4.5.1 XML

Se trata del archivo encargado de definir los objetos que aparecerán en el formulario. Este archivo ha de seguir las siguientes reglas:

El elemento raíz del archivo ha de ser una etiqueta “¡xml¿” la cual contiene todos los elementos.

Todos los elementos han de tener un atributo identificador “id” el cual bajo ningún concepto debe estar repetido, pues esto originaría errores en la ejecución de la aplicación.

Los elementos disponibles actualmente son:

- `<Label>` : Equivalente a texto. Contiene sólo un hijo, la etiqueta "`<value>`", que contiene el texto que ha mostrar.

- `<EditBox>` : Equivalente a campo de edición de texto. Contiene varios hijos. La etiqueta "`<value>`" indica el texto que contiene actualmente el campo de escritura. En el formulario modelo debería de estar vacío, mientras que adquiere valor una vez editado por algún usuario. La etiqueta "`<hint>`" muestra el mensaje de apoyo o pista para que el usuario tenga una idea de lo que ha de escribir. Este texto se mostrará en gris dentro del campo de escritura únicamente en caso de que éste esté vacío. La etiqueta "`<restrictions>`" contiene la lista de restricciones que ha de cumplir el texto ingresado en la etiqueta "`<value>`". Como ya se dijo anteriormente, estas etiquetas han de ser complementarias unas de otras pues todas han de cumplirse. Las restricciones serán definidas posteriormente.

- `<BoolBox>` : Equivalente a casilla de verificación. Contiene los hijos "`<value>`" y "`<checked>`" . La etiqueta "`<value>`" contiene el texto que acompañará a la casilla de verificación. La etiqueta "`<checked>`" indica si la casilla se encuentra marcada o no. En caso de estar marcada, el campo deberá contener el texto "`checked`" de la siguiente forma: "`<checked> checked </checked>`". En caso contrario el campo deberá estar vacío. Esto se ha hecho de esta forma para que la transformación a HTML sea inmediata.

- `<Signature>` : Correspondiente al cuadro de firmas. Contiene un hijo, `<url>`. En los formularios modelo debería estar vacío. En otros casos contiene el nombre del archivo que contiene la firma.

Restricciones:

- `<isNumber>`: corresponde con la restricción *IsNumberRes*. No tiene contenido.

- `<isInteger>`: correspondiente a *isIntegerRes*. No tiene contenido.

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<xml>
    <Label id="title">
        <value>Formulario De ejemplo</value>
    </Label>

    <Label id="parraf1">
        <value>nombre: </value>
    </Label>

    <EditBox id="cajanombre">
        <value>Pedro</value>
        <hint>Introduzca su nombre</hint>
        <restrictions>
            <minLength>4</minLength>
            <noNumbers></noNumbers>
        </restrictions>
    </EditBox>

    <BoolBox id="casilla">
        <value>¿Es usted mayor de edad?</value>
        <checked>checked</checked>
    </BoolBox>
</xml>
```

Figure 4.12: XML - Ejemplo de como programar un XML.

- `<maxValue>`: se corresponde con la restricción *MaxValueRes*. Debe contener el valor máximo que se puede escribir en el campo de escritura.
- `<minValue>`: se corresponde con la restricción *MinValueRes*. Debe contener el valor mínimo que se puede escribir en el campo de escritura.
- `<noNumbers>`: se corresponde con la restricción *NoNumbersRes*. No tiene contenido.
- `<maxLength>`: se corresponde con la restricción *MaxLengthRes*. Debe contener el valor correspondiente a la longitud máxima del valor introducido en el campo de escritura.
- `<minLength>`: se corresponde con la restricción *MinLengthRes*. Debe contener el valor correspondiente a la longitud mínima del valor introducido en el campo de escritura.
- `<IsMonth>`: correspondiente a *IsMonthRes*. Sin contenido.
- `<IsDate>`: correspondiente a *IsDateRes*. No tiene contenido.
- `<IsEmail>`: correspondiente a *IsEmailRes*. No tiene contenido.

4.5.2 XSLT

Se trata del archivo encargado de dar formato HTML a los elementos definidos en el XML. Se trata de un lenguaje estandarizado del cual hay multitud de tutoriales en la red. Sin embargo, a continuación se mostrarán algunos consejos de como usarlo adecuadamente.

El fundamento de XSLT consiste en sustituir apariciones de elementos en el archivo XML por código HTML. Por ejemplo:

- `<xsl:template match="xml"> codigo HTML</xsl:template>` ingresará en el archivo HTML el código definido dentro. Dado que la etiqueta `<xml>` es la raíz del archivo XML, sólo hay una aparición de este elemento, y puede aprovecharse para añadir

las etiquetas `<head>` y `<body>` del archivo HTML resultante, además de su contenido.

- `<xsl:value-of select="value"/>` ingresará en el archivo HTML el contenido de la etiqueta `<value>`.

- `<xsl:apply-templates select='EditBox[@id="cajaDiaPedido"]'/>` se trata posiblemente del más importante. Ésta etiqueta llamará a una plantilla definida en otro punto del archivo XSLT pero además esta plantilla sólo afectará a elementos de tipo `<EditBox>` definidos en el archivo XML cuyo atributo `id` sea `cajaDiaPedido`. Como los identificadores son únicos, aplicará sólo la plantilla al elemento deseado.

- `<xsl:template match='EditBox[@id="cajaDiaPedido"]> <u> <xsl:value-of select="value"/> </u> </xsl:template>` define una plantilla que toma el valor del campo `<value>` del EditBox cuyo identificador es `cajaDiaPedido` y lo rodea de las etiquetas `<u>` y `</u>` que hacen que se muestre este código con estilo subrayado. En caso de querer que una misma plantilla sea válida para varios elementos se separan sus nombres con el carácter `|` como puede verse en `<xsl:template match='EditBox[@id="cajaDiaPedido"] | EditBox[@id="cajaMesPedido"] | EditBox[@id="cajaDiaActual"] | EditBox[@id="cajaMesActual"]' >`.

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="xml">
  <html>
  <body>
    <xsl:apply-templates select='Label[@id="title"]' />
    <xsl:apply-templates select='Label[@id="parraf1"]' />
    <xsl:apply-templates select='EditBox[@id="cajanombre"]' />
    <br/>
    <xsl:apply-templates select='BoolBox[@id="casilla"]' />
  </body>
  </html>
</xsl:template>

<xsl:template match='Label[@id="title"]' >
            <h1>  <xsl:value-of select="value"/> </h1>
</xsl:template>

<xsl:template match='Label[@id="parraf1"]' >
            <xsl:value-of select="value"/>
</xsl:template>

<xsl:template match='EditBox[@id="cajanombre"]' >
           <xsl:value-of select="value"/>
</xsl:template>

<xsl:template match='BoolBox[@id="casilla"]' >
   <form>
        <input type="checkbox" name="{@ID}" value="{@ID}" disabled="disabled">
            <xsl:if test="checked = 'checked'">
                <xsl:attribute name="checked">checked</xsl:attribute>
            </xsl:if>
        </input>
        <xsl:value-of select="value"/>
   </form>
</xsl:template>

</xsl:stylesheet>
```

Figure 4.13: XSLT - Ejemplo de como programar un XSLT.

5

Conclusiones y Trabajo Futuro

5.1 Conclusiones

La realización de este proyecto nos ha permitido comprobar las complicaciones derivadas de un servicio de esta envergadura y su ejecución. Tanto la organización necesaria para su elaboración como los problemas surgidos para sincronizar múltiples bases de datos y usuarios, son algunos de los problemas a los que nos hemos tenido que enfrentar. Puede ser ésta una de las causas por la que en la actualidad no haya una aplicación más generalista que sirva formularios, junto con el hecho de que cada empresa quiera tener un control total sobre estos. Aún así, esta aplicación puede adaptarse según la situación lo requiera con, creemos, una mínima adaptación a la situación.

La estructura de los formularios ha sido construida con la intención de que estos puedan adaptarse fácilmente, según el cometido en el que sean requeridos. Además con la capacidad de añadir restricciones, las cuales pueden ser también muy variadas, se dota al formulario de consistencia y versatilidad a la hora de su correcto uso.

Esta aplicación además puede ser extendida con otras funciones que se detallan a continuación y que complementan y mejoran las prestaciones de la misma. Esta es una de las virtudes del proyecto, que puede ser completado con una infinidad de funciones, controlando su complejidad.

Por todo ello estamos satisfechos de esta aplicación, tanto con el procedimiento de desarrollo como con los conocimientos adquiridos durante la realización de la misma,

así como con la obtención del producto final.

5.2 Trabajo futuro

La capacidad de mejora de esta aplicación mediante el añadido de nuevas funciones es muy amplia:

- Capacidad para añadir fotos hechas con la cámara o guardadas en el dispositivo a un formulario. Con esta característica los formularios serían más útiles al contener información visual complementando los datos escritos.
- Complemento para el diseño de bocetos a mano alzada. Un pequeño complemento tipo paint para esbozar ideas sobre diseño. También podría valer para sobre una plantilla hacer cambios a mano alzada (por ejemplo señalar fallos en un mecanismo).
- Georreferenciación de los formularios. Una vez realizado esto se podría diseñar un nuevo menú para abrir formularios, posicionando en un mapa los distintos formularios para una mejor búsqueda.
- Implementación de una función de búsqueda rápida en el menú de *Abrir* y *Nuevo* y mostrar los formularios que coincidan con la búsqueda.
- Poder convertir el formulario en un PDF y poder enviarlo para imprimir a una impresora mediante Printshare.
- Poder desarrollar los formularios desde el propio dispositivo móvil y pudiendo configurar el XSLT.

6

Referencias

1. The complete Android Guide. Kevin Purdy. `http://completeandroidguide.com`.
2. Android CookBook. Ian F. Darwin. O'Reilly Media. 2012.
3. Andbook. Anddev. `www.anddev.org`.
4. Learning Android. Marko Gargenta. O'Reilly Media. 2010.
5. `www.android-spa.com`
6. `www.androidSpain.es`
7. `www.codeproject.com`
8. `www.developer.android.com`
9. `www.kaloer.com`
10. `www.mobiforge.com`
11. `www.mobile.tutsplus.com`
12. `www.sgoliver.net`
13. `www.stackoverflow.com`
14. `www.wikipedia.com`
15. `www.w3schools.com/xsl/default.asp`